# Takmičarsko programiranje – C++

## https://petlja.org/biblioteka/r/kursevi/prirucnik-cpp

### <iostream>, cin, cout, trikove za ubrzavanje citanja, ...

**Zadatak 01: cin, cout**

```
#include <iostream>

using namespace std;

int main()

{

   cout << "Zdravo svete" << endl;    cout << "Hello world" << endl;

   cout << "Bonjour monde" << endl << "Ciao mondo" << endl << "Hola mundo" << endl;

   return 0;

}
```

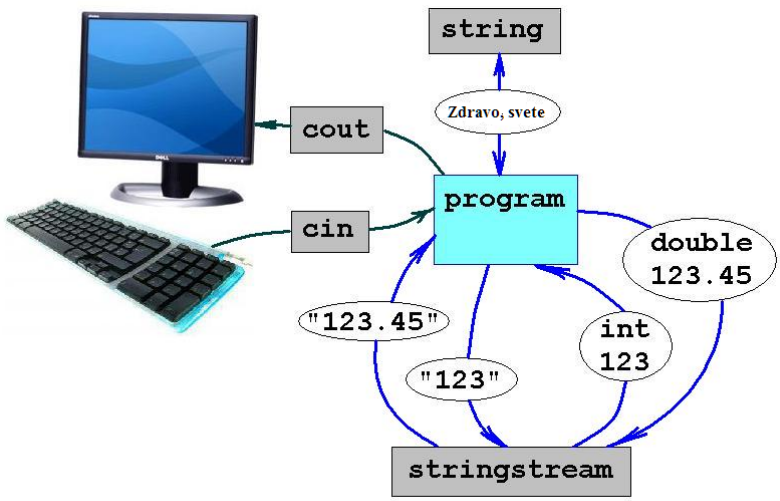
**iostream** je skraćenica od input-output stream.

Skoro svaki C ili C++ program koristi bar neku od standardnih funkcija ili klasa.

#include <iostream>  znači "uključi ulazno-izlazni tok".

U C++, kao i u sistemskom programiranju se često koristi termin tok (engleski *stream*), Ulazno-izlazni tok je sekvenca bajtova koji neprekidno teku u jednom ili u drugom smeru, u zavisnosti od toga da li se radi o izlazu

(ispis teksta na ekranu) ili ulazu (prijem teksta otkucanog na tastaturi). Bez ove linije, program za prevođenje ne bi razumeo ostatak našeg programa, konkretno liniju gde pominjemo " Zdravo svete ".




**cout << "Zdravo svete" << endl;**

Ova komanda ima tri dela: prvi, cout, koji označava standardni izlaz (standard character output - standardni tekstualni izlaz konzole i obično je to ekran računara). Drugi, operator toka "<<", koji označava da ono što sledi će biti ispisano na izlaz. Na kraju, rečenica u okviru navodnika ("Zdravo svete"), je sadržaj koji će biti ubačen/ispisan na standardni izlaz. Primeti da se komanda završava znakom ";".

Standardna biblioteka C++ koristi različite ulazno-izlazne bafere za svaku ulazno-izlaznu tehniku. Kombinovanje C++ upisa/ispisa i C-ovskih funkcije upisa/ispisa (scanf/printf) može dovesti do nepredvidivih rezultata. U ulazno-izlaznim klasama C++-a postoji funkcija ***sync_with_stdio*** koja se koristi za koordinisanje funkcija printf i scanf sa ulazno-izlaznim podacima.

**Zadatak 02: cin, cout, sinhronizacija**

`scanf`, `printf`, … parametar `format`

- `%c` `char`
- `%d` `long`, `int`
- `%u` `unsigned int`
- `%lld` `long long`
- `%f` `float`
- `%lf` `double`
- `%s` `char*`

## cin, cout sinhronizacija

Kada radimo sa fajl stream-ovima, oni su povezani sa internim baferom tipa streambuf. Ovaj bafer je memorijski blok koji se ponaša kao posrednik između toka i fizičkog fajla. Na primer, svaki put kada se pozove funkcija fputc ili putchar ili put (koja upisuje jedan karakter), karakter se ne upisuje direktno u fizički fajl, već se najprije upisuje u taj bafer. Kada se bafer isprazni, svi podaci se upisuju u fizički medijum (ako je u pitanju izlazni tok)

ili se prosto oslobađa (ako je u pitanju ulazni tok). Ovaj proces se označava kao sinhronizacija i vrši se u sledećim situacijama:
1. Prilikom zatvaranja fajla: pre zatvaranja fajla svi baferi koji još uvek nisu prazni su sinhronizovani i svi podaci iz njih se upisuju ili iščitavaju.

2. Kada je bafer pun: bafer ima ograničenu veličinu. Kada je bafer pun, automatski se sinhronizuje. 3. Eksplicitno, sa manipulatorima: kada se određeni manipulatori koriste nad tokovima, vrši se eksplicitna sinhronizacija. Ti manipulatori su: flush i endl.

4. Eksplicitno, sa funkcijom članicom sync(): pozivom funkcije sync, koja nema parametara, vrši se sinhronizacija. Ova funkcija vraća celi broj -1 ukoliko tok nema odgovarajući bafer ili u slučaju neuspeha. U drugom slučaju (ako je sinhronizacija uspela) funkcija sync vraća 0.

### ios::sync_with_stdio(true);//omogucuje mixed I/O

**Nije preporucljivo kombinovati C++ IO sa C IO, ali …**

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
ios::sync_with_stdio(true); //kombinovanje I/O
float x;
cout <<"Unesite realni broj "<<endl;
cin >> x;
printf("Realni broj na 2 decimale je %.2f\n", x);
return 0;
}
```

### Forsiranje ispisa na 5 decimala i isključivanje IO sinhronizacije

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
// iskljucujemo sinhronizaciju da bi ucitavanje teklo brze
ios::sync_with_stdio(false);
double x;
cout <<"Unesite realni broj "<<endl;
cin >> x;
cout << fixed << showpoint << setprecision(5)<< x << endl;
return 0;
}
```

**Zadatak 03: operator dvotačka u ciklusu, range-based for loops**

**/* Ucitava se jedna linija teksta.**

*** Ispiskuje se koliko se u toj liniji teksta pojavljuje kojih**

*** decimalnih cifara.**

***/**

```
#include <iostream>
#include <array>

using namespace std;

int main() {
	const int n = 10;
	int cifre[10] = {};
	//array<int,10> cifre = {};
	int ch;
	ch = cin.get();
	while(ch!=EOF && ch!='\n') {
		if(ch >='0' && ch <='9')
			cifre[ch-'0']++;
		ch = cin.get();
	}
	for(int k:cifre)
		cout << k << endl;
	return 0;
}
```

**Zadatak 04: broj PI i ispis na više decimala (unosi se P - povrsina kruga, ispisati O – obim kruga)**

```
#define _USE_MATH_DEFINES

#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main() {
  double P, r, O;
  cin >> P;
  r = sqrt(P / M_PI);
  O = 2 * r * M_PI;
  cout << fixed << showpoint << setprecision(5) << O << endl;
}
```

Osnovno pitanje u implementaciji je kako u programu predstaviti konstantu $\pi$.
U jeziku C++ postoji problem, jer ta konstanta nije standardizovana. Većina kompilatora podržava korišćenje konstante `M_PI` (po uzoru na programski jezik C), međutim, da bi se ta konstanta mogla upotrebiti, kod nekih kompilatora potrebno je na početku programa (pre `include` direktiva) navesti definiciju

```
#define _USE_MATH_DEFINES
```

i tek nakon nje uključiti zaglavlje `cmath`. Napominjemo da je navedenu definicuju u programu potrebno navesti pre svih uključivanja zaglavlja, jer se može desiti i da neko drugo zaglavlje indirektno uključi `cmath`.

Drugi način je, naravno, definisanje konstante u programu

```
const double PI = 3.14159265359;
```

međutim, postavlja se pitanje broja decimala koje je potrebno navesti (dve, tri decimale koje učenici znaju napamet nisu uvek dovoljne).

Alternativa je da se π izračuna korišćenjem nekih trigonometrijskih funkcija (npr. kao $\arccos(-1)$ ili $4 \cdot \arctan(1)$.

```
double PI = acos(-1);
```

ili

```
double PI = 4*atan(1);
```

## <vector>, indeksni pristup, push_back, resize, reserve,

```
/* Zadatak 5: Ucitava se ceo broj N, a zatim N celih brojeva
 * Ispisuje se ucitanih N brojeva obrnutim redosledom
 */

#include <iostream>
#include <vector>

using namespace std;

int main() {
	int n;
	cin >> n;
	vector<int> a(n);
	for(int i=0; i<n; i++)
		cin >> a[i];
	for(int i=n-1; i >=0; i--)
		cout << a[i] << endl;
	  //BRZE je koristiti cout << a[i] << "\n";
		 //zbog praznjenja stream-a koje radi endl

}

/*  Zadatak 06: formulacija ista kao kod prethdodnog, promena u resenju: koristi se push_back
 *
 * Ucitava ceo broj N, a zatim N celih brojeva
 * Ispisuje ispisuje ucitanih N brojeva obrnutim redosledom
 */
#include <iostream>
#include <vector>
```

```
using namespace std;

int main() {
	int n;
	cin >> n;
	vector<int> a;
	// a.reserve(n);
	for(int i=0; i<n; i++) {
		int k;
		cin >> k;
		a.push_back(k);
		// cerr << "* size=" << a.size() << " capacity=" << a.capacity() << endl;
	}
	// a.shrink_to_fit();
	// cerr << "* size=" << a.size() << " capacity=" << a.capacity() << endl;
	for(int i=n-1; i >=0; i--)
		cout << a[i] << endl;
}
```

VEKTOR STRINGOVA

```
/*  Zadatak 07: U prvoj liniji ucitava broj N i M, a zatim još N linija teksta,
 * od kojih ni jedna nije duza od 100 karaktera.
 * Ispisuje ucitanih N linija teksta, tako što prvo ispiše
 * poslednjih M linija, pa zatim prvih N-M.
 */

#include <iostream>
#include <vector>
#include <string>

using namespace std;

int main() {
	int n,m;
	cin >> n >> m;
	cin.ignore(102,'\n'); // prelazak u novi red
	vector<string> a(n);

	for(int i=0; i<n; i++)
		getline(cin, a[i]);

	for(int i=0;i<n;i++)
		cout << a[(n-m+i)%n] << endl;

}
```

```
/*  Zadatak 08: promena: koristi se printf/scanf
 *
 * U prvoj liniji ucitava broj N i M, a zatim još N linija teksta,
 * od kojih ni jedna nije duza od 100 karaktera.
 * Ispisuje ucitanih N linija teksta, tako što prvo ispiše
 * poslednjih M linija, pa zatim prvih N-M.
 */

#include <cstdio>
#include <vector>
#include <string>

using namespace std;

int main() {
	int n;
	char s[101];
	scanf("%d",&n);
	gets(s);
	int m = atoi(s);

	vector<string> a(n);

	for(int i=0; i<n; i++)  {
		gets(s);
		a[i] = s;
	}

	for(int i=0;i<n;i++) {
		printf("%s\n", a[(n-m+i)%n].c_str());
	}

}
```

## Vektor kao matrica u C++ i 1D niz u C-u

```
/* Zadatak 09: promena: pozivanje funkcije koja pretpostavlja C nizove
 *
 * U N kutija poredjanih u krug se na pocetku nalazi po M kuglica.
 * Decak se zabavlja tako što stane ispred neke kutije,
 * uzme iz te kutije jednu kuglicu i vidi koliko je u kutiji ostalo kuglica,
 * zatim se pomeri u pravcu kazaljke na satu za dvodtruko toliko broj kutija, pa
 * ceo taj postupak ponavlja dokle god je u mogucnosti.
 *
 * Ucitavaju se brojevi N i M.
 * Ispisuje se koliko je u kojoj kutiji na kraju ostalo kuglica pocevsi
 * od kutije ispred koje je decak prvo stao, pa redom u pravcu kazaljke na satu.
 */

#include <iostream>
#include <vector>
#include <cstdio>
```

```
using namespace std;

void ispisi_niz(int a[], int n) {    //C-ovski niz
	for(int i=0; i<n; i++)
		printf("%d\n", a[i]);
}

int main() {
	int n,m;
	cin >> n >> m;
	vector<int> a(n,m);
	int i=0;
	while(a[i] != 0) {
		a[i]--;
		i = (i + a[i]*2) % n;
	}
	ispisi_niz(a.data(), a.size());   // vektor se prosledjuje kao C-vski niz
}
```

## Zadatak 10:

```
/* Ucitava ceo broj N, a zatim N celih brojeva
 * Ispisuje ucitanih N brojeva, tako što ispisuje prvo
 * sve parne rastucim redosledom, a zatim sve neparne rastucim redosledom
 */

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main() {
	int n;
	cin >> n;
	vector<int> par;
	vector<int> nepar;
	for(int i=0; i<n; i++) {
		int k;
		cin >> k;
		if(k % 2 == 0)
			par.push_back(k);
		else
			nepar.push_back(k);
	}
	sort(par.begin(), par.end());
	sort(nepar.begin(), nepar.end());
	for(int k: par)
		cout << k << endl;
	for(int k: nepar)
		cout << k << endl;
}
```

## Zadatak 11: isti

```
/* promena: jedan vektor sa prilagodjenim poretkom sortiranja
```

```
 *
 * Ucitava ceo broj N, a zatim N celih brojeva
 * Ispisuje ucitanih N brojeva, tako što ispisuje prvo
 * sve parne rastucim redosledom, a zatim sve neparne rastucim redosledom
 */

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

bool poredak_za_sort(int p, int q) {
	if(p%2 != q%2)
		return p%2 < q%2;
	return p < q;
}

int main() {
	int n;
	cin >> n;
	vector<int> a(n);
	for(int &k: a)
		cin >> k;
	sort(a.begin(), a.end(), poredak_za_sort);
	for(int k: a)
		cout << k << endl;
}
```

Zadatak 12: isto

```
/* promena: koriste se iteratori
 *
 * Ucitava ceo broj N, a zatim N celih brojeva
 * Ispisuje ucitanih N brojeva, tako što ispisuje prvo
 * sve parne rastucim redosledom, a zatim sve neparne rastucim redosledom
 */
```

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

bool poredak_za_sort(int p, int q) {
	if(p%2 != q%2)
		return p%2 < q%2;
	return p < q;
}

int main() {
	int n;
	cin >> n;
	vector<int> a(n);
	for(auto poz = a.begin(); poz != a.end(); poz++)
		cin >> *poz;
	sort(a.begin(), a.end(), poredak_za_sort);
	for(auto poz = a.cbegin(); poz != a.cend(); poz++)
		cout << *poz << endl;
}
```

<string>, indeksni pristup, operator+, ...

<utility>, pair, tuple (zgodno kasnije za primere za sortiranje, a prilicno je jednostavno iako spada u strukture podataka)

iteratori - primeri na vektorima (begin, end, rbegin, rend, prev, next, ++, --, ...), nizovima, stringovima

<algorithm>, min, max (uključujući i pozive nad initializer_list), minimax, min_element, max_element, sort, binary_search, lower_bound, upper_bound, equal_range, find, find_if, copy, copy_if, transform, count, count_if, ...

Na primerima sort i binary_search & friends pokazati moguće načine prosleđivanja funkcije poređenja (sa naglaskom na lambda funkcije).